A Hybrid Model for Application Development and Deployment Based on the Multi-tier Architecture

Liming Ma[1, 2], Xiao Ma[1], Sanxing Cao[2]

1. Information Engineering School; 2. New Media Institute
Communication University of China,
Beijing 100024, China
maliming@cuc.edu.cn

Keywords: Hybrid Model; Multi-tier Architecture; WebKit; JavaScript Engine; Extension;

Abstract

This paper proposes a new hybrid model based on the multi-tier architecture, which combines the advantages of the B/S model and C/S model in the development and deployment of application. It focuses on the implementation of the Third-party Service Provider and the Presentation tier, interspersing with the introduction of the technology selections of this model. Finally discussed is the performance optimization by utilizing the V8 JavaScript Engine in WebKit.

1 Introduction

B/S (Browser/Server) model, as well as C/S (Client/Server) model, is the mainstream application development and deployment model at present. They have similarities when the Browser is regarded as the special kind of Client. From the viewpoint of software engineering, they are both built on the basis of the multi-tier architecture [1], in which the presentation, the application processing and the data management are logically separated processes. The three-tier architecture [2] is the most widespread use of multi-tier architecture, which consists of a Presentation tier, a Logic tier and a Data tier, which is intended to allow any of the three tiers to be upgraded or replaced independently in response to the changes in requirement. Therefore this architecture is widely applied both in the B/S model and C/S model. Figure 1 shows the structure diagram of the three-tier architecture in B/S model.

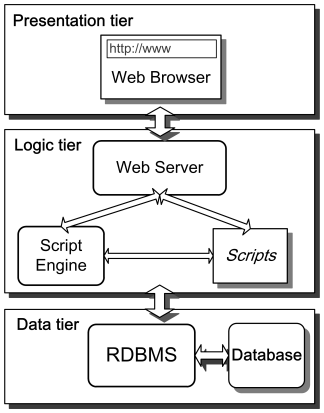

Figure 1. The Three-tier Architecture in B/S Model

However, it's obvious that there're core differences between the two models. The issue is focused on the controversy of the Client's functional boundary, which is differentiated into Thin Client (Browser) and Rich Client. Normally, Browser as the Thin Client is just responsible for loading the resources sent from the Web server, making them rendered correctly and handling the user's interaction by the browser-supported language such as JavaScript. The rest of the work will be transferred to the Server-side, which can utilize the computing capabilities of servers to execute the procedure for processing information and accessing to proprietary databases or other data sources according to the specific service logic. The inherent support for cross-platform compatibility and the ubiquity of web browsers enable to update and maintain application based on B/S model without distributing and installing software on potentially thousands of computers. The typical Rich Client is a stand-alone executable program with a graphical interface composed of several controls for the user. It has a fast response performance because it can invoke more local resources through the operating system. But it also has the weak point of a relatively high maintenance costs when facing the challenge of upgrading because there is so much logic embedded in it.

With the advent of the advanced Web applications, Browser starts to undertake more and more heavy workload. At the same time, some applications running in the Browser require the adequate access to local computing capabilities and resources to meet the specific demands, comparing favorable with the application based on C/S model. To meet the requirements above, we put forward a new type of hybrid model based on the multi-tier architecture, which is composed of a Presentation tier based on WebKit [3], a Third-party Service Description tier that is a newly added tier, a Logic tier and a Data tier. It is noteworthy that the Third-party Service Provider tier is the innovation points of this hybrid model. It's a middleware for guiding the communication between the Presentation tier and the Logic tier. All the compositions mentioned above constitute a multi-tier architecture as Figure 2 shows.

In contrast to Figure 1 and Figure 2, it can be found that there's an intuitive modification of the Presentation tier in hybrid model, that is the address bar has been discarded. The newly added Third-party Service Provider tier manages which application the Presentation tier presents and which extensions the Presentation tier requires. The specific implementation will be described in the follow sections.

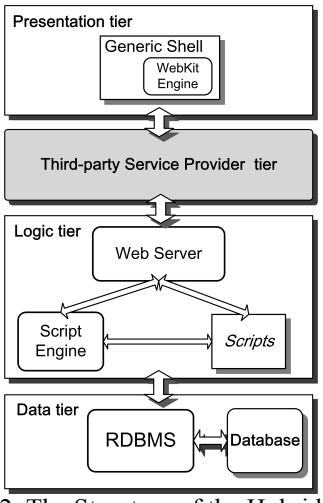

Figure 2. The Structure of the Hybrid Model

2 Logic Tier and Data Tier

The Logic tier and Data tier in the hybrid model are similar to the ones in the B/S model, acting as the brain to the application. Web Server and Database Server are respectively in the two tiers. The Web Server contains the logic or service rules corresponding to the application. Its data requirements are satisfied when it acts as a client to the Database Server who provides the data access logic and handles the storage, retrieval, and modification of data, while verifying data integrity. This is most often accomplished with the RDBMS that becomes a server to the Web Server [4].

The mainstream Web Server including Apache, IIS, lighttpd and nginx realizes the application logic and processes database data by integrating the Server-side programming language operating environment. There are several selections for Server-side programming such as PHP, Python, Ruby on Rails, .NET and Java. Some of the languages are compiled, others are interpreted, and it mainly depends on the tradeoff between performance and development costs. In general, the performance of Compiled Language is better than Interpreted Language, but the development and maintenance costs of the former are higher. Therefore, the Interpreted Language such as PHP, Python or Ruby on Rails is more suitable for rapid development and iteration, while the Compiled Language, such as C# (.NET) or Java, is widely applied in financial and the other field sensitive to performance and security.

The instances of Database Server are Oracle, DB2, Microsoft SQL Server that are commercial except for the open source database servers such as MySQL and PostgreSQL. Among all the selection of Database Server, MySQL is the best choice to support the query-based applications that read data from the database frequently. But if there is the demand to utilize the transaction processing and other advanced functions, it is tended to Oracle, DB2 or Microsoft SQL Server.

Web Server and Database Server coordinating with the operating system make up the complete environment. A famous collocation is called LAMP, referring to the first letters of Linux (operating system), Apache HTTP Server, MySQL and PHP (or sometimes Perl or Python).

3 Third-party Service Provider Tier

The initial Presentation tier, a pure and universal presentation framework based on WebKit, won't take effect until finish configuring and attaching extensions. The section of Presentation-tier Based on WebKit below will elaborate the technical details of WebKit. For the Presentation tier in hybrid model, an extra tier is demanded to guide the communication between the Presentation tier and the Logic tier. Therefore the so-called Third-party Service Provider (hereinafter referred to as TPSP) is implemented to manage all the configurations for applications and the subsequent update notification. Application Routing Map and Extensions Repository are the main functions of TPSP. Figure 3 shows the operating schematic of TPSP.

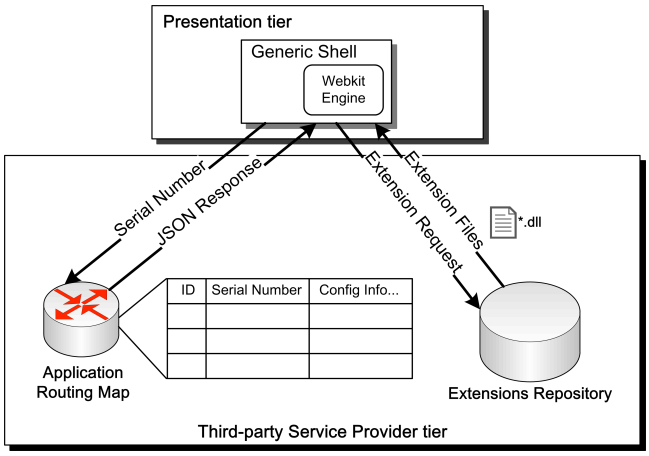

Figure 3. The Operating Schematic of TPSP

3.1 Application Routing Map

Application Routing Map is responsible for maintaining a relationship map table, which reflects the corresponding relationship of the Serial Number, a unique string that TPSP assigned to applications previously to identify each other, and the information of application's name, icon, access address. Users are prompted to input a Serial Number at the step of client's installation. Hereafter, the installer will initiate a request taking along the Serial Number to TPSP and initialize the configuration file by parsing the response, a JSON (JavaScript Object Notation) formatted text. Different applications have different Serial Numbers and configuration information, however they have the same Presentation tier that is a generic shell for the applications in different areas, simplifying the difficulty of customized development.

3.2 Extensions Repository

The functionality of initial Presentation tier is limited, while we can enhance it by the Extension Mechanism that will elaborate below. Extensions are developed and attached to the Presentation tier to make it qualified. Extensions Repository is a trusteeship cloud for the extensions to provide services on two aspects, namely extension management and extension distribution. All the extensions are stored in the repository previously whether general or dedicated, and then associated with the applications. In the response of Application Routing

Map, there's the information describing the dependency of required extensions besides what have been introduced above. By the guidance of response information, the installer fetches the extensions from Extensions Repository to the client of Presentation tier. Furthermore, the subsequent version upgrade is also in a similar approach to receive the response and fetch the new extension.

4 Presentation Tier Based on WebKit

4.1 The Basis of WebKit

The WebKit, which is the kernel to execute the application based on B/S model with the advantages of clear, stable, efficient, maintainable and compatible for the Web standards, is adopted as the foundation of the Presentation tier in the hybrid model with some alterations. In the sight of WebKit project, it's composed of WebCore and JavaScript Engine. WebCore is a layout, rendering and Document Object Model (hereinafter referred to as DOM) library for HTML and SVG. JavaScript Engine contains the Script Interpreter, Analyzer and Execution Procedure, supporting to interpret and execute JavaScript (or ECMAScript). By studying and transforming the WebKit and its core components, we integrate it into the Presentation tier client as a generic shell for hosting all the application based on the hybrid model. Figure 4 shows the composition of WebKit, describing the relationship among the components of WebKit.

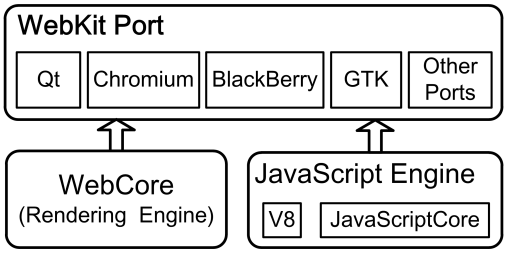

Figure 4. The Composition of WebKit

There're two issues helping us understand the operating principle of WebKit.

The first issue is how does WebKit create the GUI (graphical user interface). In general, WebKit parses the HTML file, and then renders the resources gotten from Back-end. There're two loading pipelines in WebKit, one for loading documents into frames and another for loading the subresources (such as images and scripts). WebCore is responsible for parsing the document and creating a DOM tree after the frame finishes loading the document. And there is a render tree one-to-one correspondence to the DOM tree, when nodes are created and added to the render tree, they have no position or size yet. All of the nodes have their positions and sizes determined in the process that is called layout. Since then, the GUI is rendered in the client.

The second issue is how does WebKit handle the user interaction. JavaScript Engine is the core to achieve this goal. However, there're different implementations of JavaScript Engine such as JavaScriptCore (hereinafter referred to as JSC), V8 [5], etc. JSC is a mature JavaScript Engine for WebKit, which is adopted by Apple WebKit, QtWebKit and other branches of the WebKit project except for Chromium. V8 is an open source JavaScript Engine developed by Google and applied to the Chromium project, which increases performance by compiling JavaScript to the native machine code. WebKit project takes an approach called binding to adapt the different implementations of JavaScript Engine. There's a standard set of IDL (Interface Description Language), which is a practical and useful tool for controlling the exchange of structured data between different components of a large system. It is principally designed to provide developers with a means to be able to specify interfaces to objects independent of their implementation [6]. In the stage of project file generation, According to the IDL, Python and Perl scripts generate the codes for different implementations. In the below section of this paper, we will further compare the two implementations of JavaScript Engine.

4.2 Extension Mechanism

Extension Mechanism is a technical approach to extend the JavaScript Engine for enhancement. The main target is to improve the ability of Presentation tier to deal with complex application requirements, which the native JavaScript is not qualified to meet.

An obvious approach is to implement the function module and add some additional functionality in the kernel of the WebKit directly, and then bind module to the variable linked list on the stage of global object initialization in JavaScript Engine. The extension module can be directly accessed in JavaScript code by invoking the variable of global object. However, this approach has a poor reusability when upgrading the kernel. Except for a few of underlying modules, the others should be implemented by a specific mechanism, which keeps the integrity and independence of WebKit kernel code.

The large volume of the WebKit kernel code has led to a high cost of understanding and modification. The direct approach of source code modification for extension is not feasible for all situations. According to the scalability theory of software engineering, the extension mechanism should be flexible and scalable. Each extension is mapped to a JavaScript Object, mounted under the window object (the root of the DOM tree) just like the native Objects. We can conclude this approach as Object Mapping, and there're four necessary conditions as follows.

(1) The mapping Object should be mounted to the window object.

(2) It should implement the prototype object.

(3) It should provide the constructor function.

(4) The member variables and functions of the mapping object should be adapted to the varied JavaScript Engines (JavaScript Engine adaptability).

Besides the Object Mapping, Variable Mapping is another simplified approach to extend JavaScript. QtWebKit, a relatively mature WebKit wrapper library, provides some JavaScript interfaces to implement the mapping between the QObject object of Qt and the JavaScript variable by mounting the object pointer of QObject to the global variables in JavaScript interpreter. Assuredly, the drawbacks on Variable

Mapping are also conspicuous. QtWebKit is dependent on the Qt library that is too large for some lightweight applications. And this approach is not flexible enough because it doesn't meet the condition 2 and condition 3 mentioned above.

4.3 Performance Optimization

The efficiency of JavaScript Engine is the key factors affect the performance of the application, it is important to understand the architectural execution features of JavaScript benchmarks, given performance deficiencies of JavaScript codes due to bytecode interpretation [7]. The mainstream JavaScript Engine includes JSC in WebKit, V8 in Chromium, Chakra in IE 9.0, Carakan in Opera 12.0, JaegerMonkey in Firefox 13.0.1, etc.

We have tested the performance parameters of the above-mentioned JavaScript Engine, including OS Kernel Simulation Benchmark (Richards), One-way Constraint Solver (DeltaBlue), Encryption & Decryption Benchmark (Crypto), Ray Tracer Benchmark (RayTrace), Classic Scheme Benchmark (EarleyBoyer), Regular Expression Benchmark generated by extracting regular expression operations from 50 of the most popular web pages (RegExp), Data Manipulation Benchmark that deals with splay trees and exercises the automatic memory management subsystem (Splay), Solving Navier Stokes equations in 2D, heavily manipulating double precision arrays (NavierStoke). The final score is computed as the geometric mean of the individual results to make it independent of the running times of the individual benchmarks and of a reference system.

The test environment is as follows.

- Operating System: Windows 7 with SP1 32-bit
- CPU: Intel Core 2 Duo T6670 2.20GHz
- Memory: 3GB

Table 1 shows the item scores and total score of QtWebKit with JSC, QtWebKit with V8, Chromium 20, Safari 5, Opera 12, IE 9 and Firefox 13.

	QtWebKit with JSC	*QtWebKit with V8*	*Chromium (20.0.1132.57)*	*Safari (5.1.7)*	*Opera (12.0)*	*IE (9.0)*	*Firefox (13.0.1)*
Richards	2148	13274	9058	2030	2567	1413	5920
DeltaBlue	1491	17316	11696	1744	2034	1501	6182
Crypto	2390	16415	9026	2581	3159	2869	9514
RayTrace	2349	20202	10372	2922	3256	1145	1954
EarleyBoyer	2000	28521	14935	2505	3208	2427	4563
RegExp	593	3640	1921	865	1013	1366	835
Splay	1490	5127	2355	2249	5264	1115	6617
NavierStokes	1785	18587	8738	2092	2762	1540	8738
Total Score	**1657**	**12871**	**7014**	**2015**	**2664**	**1584**	**4436**

Table 1: Table of Benchmark Test Results

Figure 5 shows the comparison of benchmark test in totoal score, which reflects the result that V8 JavaScript Engine is more efficient than others.

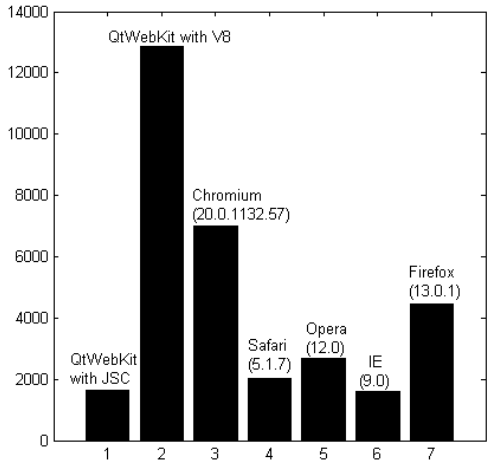

Figure 5. Bar Chart of Benchmark Test Results

By comparing the performance of various JavaScript Engine, we finally pick out V8 to support the Presentation tier, which implements ECMAScript as specified in ECMA-262, 5th edition, compiles and executes JavaScript source code, handles memory allocation for objects, and garbage collects objects it no longer needs. And it enables any C++ application to expose its own objects and functions to JavaScript code.

5 Conclusion

This paper proposes a new hybrid model based on the multi-tier architecture, which combines the advantages of the B/S model and C/S model in the development and deployment of application and simplifies the difficulty of customized development for applications in different areas. It not only maintains the features of low cost of deployment, agile iterative upgrading, cross-platform support, and other advantages on B/S model, but also can access to the local computing capabilities and resources adequately just like the application on C/S model.

In this new hybrid model based on the multi-tier architecture, we add a Third-party Service Provider tier to guide the communication between the Presentation tier and the Logic tier. WebKit is adopted as the foundation of the Presentation tier with some alterations, which includes the extension mechanism and the performance optimization.

We hereby hope that the hybrid model proposed in this paper could act as an effective reference for the application development and deployment of this kind.

Acknowledgements

This work is sponsored by the 382 Talents Plan (G08382304), Chinese National Hi-tech Industrialization Project (CNGI-09-03-15-1), and the Virtual Library Technology project granted by China Art Science and Technology Institute.

References

[1] Hans-Peter Steiert. "Towards a component-based n-Tier C/S-architecture", *ISAW '98 Proceedings of the third international workshop on Software architecture*, pp. 137-140.

[2] Eckerson, Wayne W. "Three Tier Client/Server Architecture: Achieving Scalability, Performance, and Efficiency in Client Server Applications", *Open Information Systems*, Vol. 10, Nr. 1, (1995).

[3] "The WebKit Open Projects", http://www.webkit.org/

[4] Alexander Pons, "Evaluation of Server-side Technology for Web Deployment". *Issues in Information Systems (IIS)*, Vol. 04, 2003, pp. 667-671.

[5] Introduction of Google V8 JavaScript Engine, https://developers.google.com/v8/intro.

[6] David Alex, Lamb Queen's Univ., Kingston, Ontario, Canada. "IDL: sharing intermediate representations", *ACM Transactions on Programming Languages and Systems (TOPLAS)*, Volume 9 Issue 3, July 1987, pp. 297-318.

[7] Tiwari, D. "Architectural characterization and similarity analysis of sunspider and Google's V8 Javascript benchmarks", *Performance Analysis of Systems and Software (ISPASS)*, Date of Conference: 1-3 April 2012, pp. 221-232.

[8] Bjorn N. Freeman-Benson, John Maloney. "The DeltaBlue Algorithm: An Incremental Constraint Hierarchy Solver", *January 1990 Communications of the ACM*.

[9] Watkins, S. Fac. of Inf. Technol., Monash Univ., Clayton, Vic. Dick, M.; Thompson, D. "From UML to IDL: a case study", *Technology of Object-Oriented Languages*, 1998. TOOLS 28. Proceedings, 23-26 Nov 1998, pp. 141-153.

[10] Jiang Zhang-gai, Chen Rong. "Native Extension Strategy of WebKit Based on CAR Components", *Journal of Computer Applications*, 2009, 29(z2).

[11] C. Chambers, D. Ungar, and E. Lee. 1989. "An efficient implementation of SELF a dynamically-typed object-oriented language based on prototypes", *SIGPLAN Not*, 24, 10 (September 1989), pp. 49-70.